Native WiFi Backscatter

Longzhi Yuan, *Member, IEEE*, Wei Gong, *Senior Member, IEEE*, and Yuguang Fang, *Fellow, IEEE, ACM*

***Abstract*— WiFi backscatter has attracted intensive attention because the large population of WiFi radios can provide plenty of excitation signals. However, WiFi backscatter communication has imposed unwanted constraints on either exciters or receivers since its inception. In this paper, we present Chameleon, a native WiFi backscatter system, where WiFi tags can generate native WiFi packets using uncontrolled productive WiFi signals as carriers. Our tag-only design requires no particular excitation patterns and no changes in software/hardware on WiFi network interface cards (NICs). The key idea is for the Chameleon tag to demodulate the productive WiFi signal and backscatter it into a full-function packet using on-the-fly modulation. To align tag decoding and modulation with excitation symbols, we design a time synchronization and clock compensation scheme suitable for low-power tags. We prototype WiFi tags using ultra-low-power FPGAs and evaluate them in real-world scenarios where excitations are ambient traffic and backscatter receivers are a wide range of commercial off-the-shelf (COTS) NICs. Comprehensive field studies show that the maximal backscatter throughput of Chameleon is almost 1 Mbps, which is over 125× and 1000× higher than what WiTAG and FS-Backscatter tags could achieve, respectively. We also show that Chameleon can natively communicate with various COTS WiFi devices on Windows, iOS, and Android platforms. We believe that this design will enable ubiquitous WiFi connectivity for billions of IoT devices via widely available mobile gadgets and existing wireless infrastructure.**

***Index Terms*— Internet of Things (IoT), backscatter, low power.**

I. Introduction

WiFi backscatter has attracted tremendous attention from both academia and industries [1]. Its key advantages over backscatter systems with other commercial radios include the existence of a large population of WiFi radios and frequent packet exchanges between WiFi devices. There are over 3 billion WiFi devices globally shipped out each year [2], and every WiFi radio generates significantly more packets than other commercial radios such as Bluetooth and ZigBee. There are much more abundant WiFi packets around us, providing more ample excitation signals for WiFi backscatter than any

Manuscript received 19 October 2023; revised 10 March 2024; accepted 3 May 2024; approved by IEEE/ACM Transactions on Networking Editor C. Peng. The work of Longzhi Yuan and Yuguang Fang was supported in part by Hong Kong SAR Government through the Global STEM Professorship and Research Talent Hub and in part by Hong Kong Jockey Club through the JC STEM Lab of Smart City. The work of Wei Gong was supported in part by NSFC under Grant 61932017 and Grant 61971390. *(Corresponding author: Yuguang Fang.)*

Longzhi Yuan and Yuguang Fang are with the Department of Computer Science, City University of Hong Kong, Hong Kong, China (e-mail: longyuan@cityu.edu.hk; my.fang@cityu.edu.hk).

Wei Gong is with the School of Computer Science and Technology and the School of Data Science, University of Science and Technology of China, Hefei 230026, China (e-mail: weigong@ustc.edu.cn).

This article has supplementary downloadable material available at https://doi.org/10.1109/TNET.2024.3409081, provided by the authors.

Digital Object Identifier 10.1109/TNET.2024.3409081

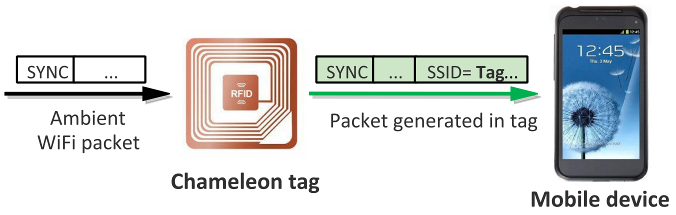
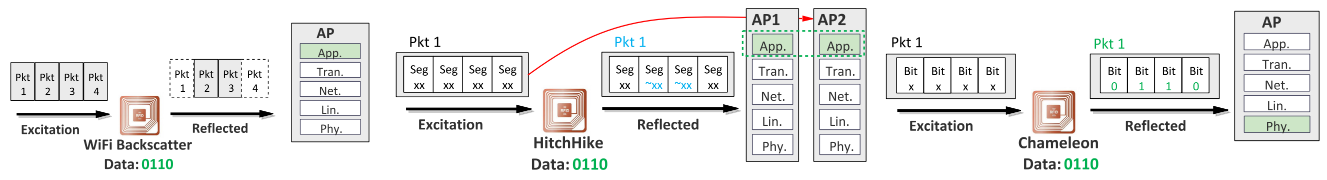


Fig. 1. Conceptual design of Chameleon that generates native WiFi packets out of uncontrolled ambient WiFi signals, ensuring 100% compatibility with COTS WiFi devices.

other backscatter communication systems. If WiFi backscatter tags can be embedded into daily-used objects, such as health monitoring devices, books, and even pills, and connect to the Internet with the help of massively deployed WiFi devices, human activities can be managed uniformly online, and the vision of pervasive connectivity for everything [3] could soon become a reality. Unfortunately, existing WiFi backscatter systems are unable to leverage ambient packets as excitation efficiently [4], [5], [6], [7], [8], [9], [10], [11], [12], [13], [14], [15], [16]. The main difficulty is that they cannot recognize random bits in ambient packets. As a result, they can only modulate those WiFi packets blindly. However, limited by the low-power and low-cost promise of backscatter, blind modulation is not enough to reshape packet content and keep only tag bits. Thus, tag bits and excitation bits are mixed in backscatter packets. For ease of analysis, we categorize existing WiFi backscatter systems into two generations. The first generation [1], [5], [11] adopts packet-level ASK modulation based on Received Signal Strength Indicator (RSSI) by reflecting or absorbing packets. This idea does work, but the so-designed systems all suffer from poor throughput. This is mainly because they use a whole packet to convey only one tag bit. What is worse, because ambient WiFi packets are sporadic and intermittent, the receiver may have difficulty in distinguishing whether a tag or an exciter causes RSSI change. The second generation attempts to achieve high throughput using symbol-level backscatter and introduces an additional receiver to obtain productive data from wireless carriers [8], [9]. These systems significantly improve backscatter throughput, but need two synchronized receivers [14], [17], and thus tend to experience high communication instability [13]. Application scenarios are still far from those as shown in Fig. 1.

Ideally, we expect that a tag can modulate ambient packets, and then a single COTS WiFi NIC is adequate to demodulate tag data from a backscattered packet alone, the same way as to demodulate an active WiFi packet, as shown in Fig. 1. So we call a backscatter system WiFi native if a WiFi NIC does not need to distinguish active or backscattered packets



(a) 1st generation WiFi backscatter. (b) 2nd generation WiFi backscatter. (c) Our proposal.

Fig. 2. Comparison of three generations for WiFi backscatter, where our proposal is the first native WiFi backscatter system.

and achieves tag data transparency on the PHY layer. Under this criterion, we compare existing WiFi backscatter systems with our work. As shown in Fig. 2, the first-generation WiFi backscatter systems are not native since they have to demodulate tag data based on RSSI by writing an App on the application layer [1], [5]. The second-generation systems are not native either because they need to perform reverse codeword translation either on the application layer [8], [9] or on the modified WiFi PHY layer [14].

All prior WiFi backscatter systems lose support for native WiFi because their approaches are indirect and they have not focused on the common drawbacks such as varying excitation signals and simple backscatter actions, as introduced above. The first-generation works skip this problem by manipulating excitation at the packet level. The second-generation works postpone these problems to the receiver side. They modulate an excitation signal in symbol but deploy an additional dedicated receiver to get the content of the excitation packet. They could not overcome the problem of varying carrier signals, either. In contrast, in this paper, we address this fundamental problem directly. Specifically, we demodulate ambient 802.11b WiFi signals at the tag, removing the negativity brought by content-varying carriers once and for all. More specifically, we introduce the first third-generation WiFi backscatter system, Chameleon, that can modulate varying 802.11b carrier signals into native WiFi packets with only tag bits.

As aforementioned, previous ambient WiFi backscatter systems are not native, and all the means to generate native WiFi packets require single-tone continuous wave (CW). On the contrary, Chameleon leverages the following bold idea: If a tag can demodulate a productive carrier, the whole carrier will become a virtual CW. Thus, we only need to modulate the difference between the productive data and the target tag data. Doing so brings several unprecedented advantages. 1) *Native*. Our backscattered packet is of full function since every symbol is under control. The packets from prior systems, e.g., Hitchhike and Freerider [8], [9], have intrinsic cyclic redundancy code (CRC) errors, which means those packets cannot pass the MAC layer because they will yield integrity check value (ICV) errors, indicating verification failures for all WEP and WPA-related communications. 2) *Transparent*. A single WiFi NIC can receive our backscattered packets without the need of even software change because our packets are native from the PHY layer to the application layer. 3) *Fast*. The backscattered packets can achieve the same rate as the WiFi carrier signals.

Repurposing ambient content-varying carriers is challenging for two main reasons. 1) *Passive Demodulation for WiFi*. Although recent advanced systems introduce several novel designs for backscatter downlinks, none of them can demodulate productive WiFi data. MIXIQ [18] does not work with random content-varying WiFi because it requires an intelligent (constrained) helper signal. Saiyan [19] introduces a frequency demodulation method and thus does not fit for phase-modulated WiFi. Passive DSSS [20] is a close-loop system where both excitaters and receivers are specially dedicated and incompatible with commodity WiFi. Unlike these works, we propose demodulate WiFi signals based on the key observation that phase-modulated WiFi signals can be differentiated by their pulse widths, which is detailed in Section II-B. 2) *Backscatter Modulation*. To support native WiFi backscatter, it requires a novel modulation based on the carrier and tag data at the same time. However, all prior backscatter modulation methods depend only on tag data [8], [15]. As a result, we design an on-the-fly modulation that creatively backscatters the difference between tag data and carrier data, ensuring the backscattered packets 100% native.

To ensure high-quality demodulation, we also design a novel template-matching-based synchronization to achieve accurate symbol timing. With the fact that the backscatter tag clock may have serious errors, we enable it to lock on the excitation and improve the error tolerance from below $\pm 20ppm$ to $\pm 2000ppm$. Moreover, we build tag prototypes using low-power FPGAs and IC simulation. Various real-world tests show that Chameleon realizes an uplink throughput of 992.4 kbps, which is over 125$\times$, over 1,000$\times$, and over 100,000$\times$ higher than WiTAG, FS-Backscatter, and WiFi-Backscatter, respectively. Further, we show that the backscattered signals of Chameleon can be natively supported by a wide range of COTS WiFi devices, including different laptops, tablets, and smartphones. Currently, only differential binary phase shift keying (DBPSK)-802.11b WiFi is supported in Chameleon. This is because different pulse width patterns for bit ‘0’ and bit ‘1’ only exist in DBPSK-802.11b WiFi signals. In summary, our technical contributions are summarized as follows:

- We propose the first passive demodulation, Chameleon, for 802.11b WiFi systems, which is to use pulse widths to distinguish WiFi symbols.
- We design a novel on-the-fly modulation that can turn arbitrary 802.11b WiFi carriers into native WiFi packets.
- The time synchronization and clock compensation enable Chameleon to work in real IoT scenarios and maintain good compatibility with most COST WiFi systems.

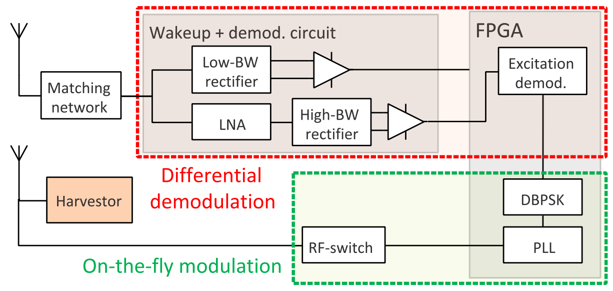


Fig. 3. Chameleon overview.

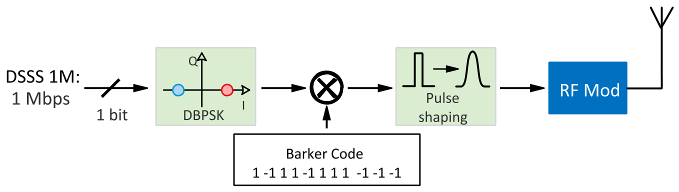


Fig. 4. 802.11b WiFi signal generation.

II. Schematic Design

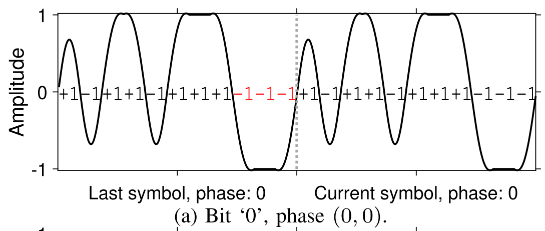
A. Overview

As shown in Fig. 3, Chameleon mainly consists of the energy harvesting module, differential demodulation, and on-the-fly modulation. Differential demodulation detects ambient RF signals, finds ongoing packets, and then wakes up the system using a wakeup circuit composed of a low-bandwidth rectifier and a comparator. Afterward, a high-bandwidth rectifier extracts the binary envelope and feeds it to the demodulation algorithm on an FPGA. The carrier protocol can be identified using the envelope-based method introduced in Multiscatter [13]. Whether it is the target DBPSK-802.11b WiFi signal is determined in the DBPSK-demodulated physical header, which can be decoded in Chameleon. While the carrier data is being demodulated, the tag modulates tag data together with the demodulated carrier data on the fly. This enables tag to reshape the old WiFi packet into another native WiFi packet. The harvester module is embedded into our battery-free prototype to be introduced in Section. III to harvest from ambient light and RF signals.

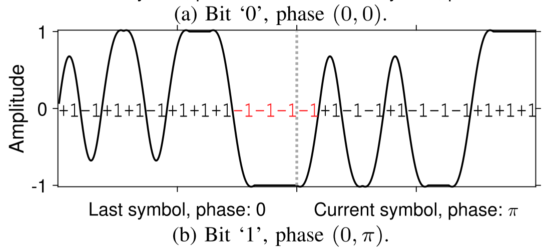
B. Observations of Symbol Pulses

According to WiFi standards, the symbols of 802.11b WiFi packets are phase-modulated. So the common wisdom for demodulation is to extract the phases of symbols through a mixer, which is usually power-hungry for high-frequency signals. Instead, in our scheme, we seek to use a low-power envelope detector to accomplish this job. Specifically, we will analyze how WiFi signals are generated on the transmitter side and what those signals would be like after an envelope detector on the tag side. The procedure is shown in Fig. 4.

At the WiFi transmitter, a series of raw data bits are first modulated using DBPSK, then spread with an 11-bit barker code sequence. After that, it is processed using a shaping filter, e.g., Gaussian filter and root cosine filter, to remove unwanted high frequencies. During DBPSK modulation, raw data bits '0' and '1' are mapped to phase differences of 0 and π between two adjacent symbols. To modulate a bit '0', the phase of the current symbol will be exactly the same as the last one, e.g., $(0, 0)$ in Fig. 5a. Similarly, a bit '1' can be mapped to two symbols whose phases differ by π, such as $(0, \pi)$ in Fig. 5b.



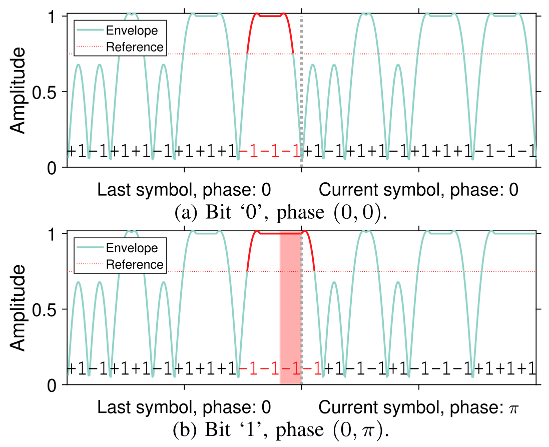
(a) Bit '0', phase $(0, 0)$.



(b) Bit '1', phase $(0, \pi)$.

Fig. 5. Waveforms of two consecutive symbols on the transmitter side.

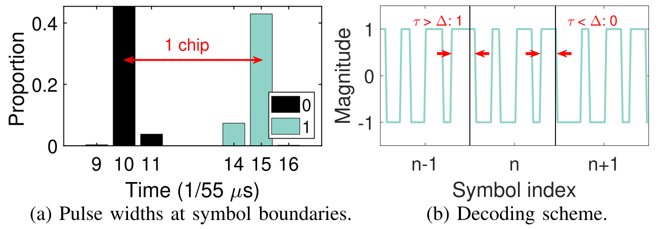
After DBPSK modulation, the transmitter performs DSSS to spread its signals, which is designed for interference suppression. In the WiFi standard, this spreading sequence is an 11-bit Barker code, which is predefined as $[+1-1+1+1-1+1+1+1-1-1-1]$. This operation divides a WiFi symbol into 11 chips, each of which lasts for $\frac{1}{11}$ μs. Along with the spreading process, we observe that if the raw bit is '0', there are three identical chips around the symbol boundary when the phases are $(0, 0)$ or (π, π). The former is shown in Fig. 5a, and the latter can be found in the *Section I* of *Supplementary Material*. On the other side, if the raw bit is '1', we always observe four identical chips at the symbol boundary. The waveform with phases $(0, \pi)$ is shown in Fig. 5b. Still, the phases can also be $(\pi, 0)$, whose waveform is shown in the *Section I* of *Supplementary Material*. Such observations motivate our idea: if we can tell the differences between the pulse width at the symbol boundary, deducing the raw data bits is made easy. Let us continue to examine the transmission process and see whether the above features can be observed. The next important procedure is the shaping filter. We adopt a common Gaussian filter to check its effects. As shown in Fig. 5, the change from $+1$ to -1 or from -1 to $+1$ just becomes smooth and does not destroy pulse-width features for different raw data bits. Obviously, the signal envelope is not constant anymore. Therefore, we believe that it is possible to demodulate this WiFi signal by measuring the pulse widths at symbol boundaries. To verify this idea, we design a passive rectifier to extract the baseband envelope. Part of the acquired envelopes at the tag are shown in Fig. 6. It can be seen that the high duration of phases $(0, \pi)$ (bit '1') is longer than that of phases $(0, 0)$ (bit '0'). Such phenomenon also appears in other cases, which can be seen in *Section I* of *Supplementary Material*. The additional identical chip causes such a difference as the transmission process we analyze previously. Specifically, this difference equals the duration of a single chip, $\frac{1}{11}$ μs. To further corroborate this, we use



(a) Bit '0', phase $(0, 0)$.

(b) Bit '1', phase $(0, \pi)$.

Fig. 6. Envelopes of two consecutive symbols on the tag side.



(a) Pulse widths at symbol boundaries.

(b) Decoding scheme.

Fig. 7. We measure pulse widths at symbol boundaries for random 802.11b WiFi signals. Results show that the pulse widths of '0' and '1' are distinguishable. The decoding can be as easy as finding symbol boundaries and checking pulse width is great enough.

a low-power comparator to digitize the rectifier output and process the sampling results in FPGA. Through extensive evaluation, we plot a subset of our empirical measurement results shown in Fig. 7a, which contain 1000 packets, each with a 100-byte payload. From the figure, we can observe that two clusters are naturally formed, of which the centers are $\frac{10}{55}$ μs and $\frac{15}{55}$ μs, respectively. The distance between the two centers is exactly $\frac{1}{11}$ μs.

After reviewing the transmission process and WiFi envelopes at the tag, we conclude that we are able to demodulate WiFi signals at the tag by examining envelope differences. The key idea is to convert 802.11b DSSS signals using barker codes into signals with differences on symbol boundaries, i.e., a raw bit '1' has a discernible longer pulse width than a raw bit '0', which achieves the first passive WiFi demodulator. It is worth noting that the phase of the last symbol is needed as a reference for the current symbol to realize DBPSK demodulation in the active 802.11b receiver [21]. However, our passive decoding directly observes the envelope pattern for the phase difference between consecutive symbols and thus does not need to know the last symbol.

C. Differential Demodulation With Accurate Timing

In the previous discussion, we demonstrated that passive WiFi demodulation is possible. As shown in Fig. 7, the specific solution is to find the symbol boundary and then check if the pulse width τ is greater than the threshold τ_δ. If yes, the symbol is '1'. Otherwise, it is '0'. τ_δ can be chosen as the mean value of pulse widths corresponding to '1' and '0'.

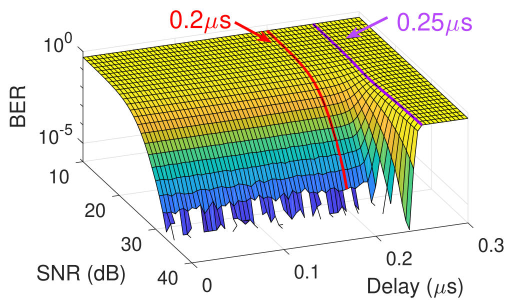
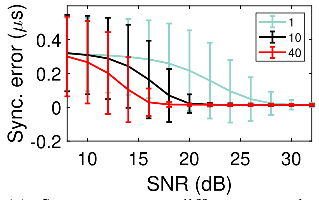


Fig. 8. Downlink demodulation BER vs controlled sync errors.

However, how to find the symbol boundaries accurately at the tag is crucial for the performance of our proposed scheme. We first examine how accurate symbol synchronization we need to achieve and then design a low-power scheme using FPGA implementation to fulfill the requirements.

1) Timing Requirements: For synchronization requirements, we conduct simulations in *Matlab* to investigate demodulation BERs under different controlled synchronization errors. Fig. 8 shows the relationship between demodulation BER and the timing delay. As can be seen, the delay below 0.2 μs has a negligible impact on demodulation quality. However, when it is more than 0.25 μs, the demodulation is bound to fail. To conclude, when the delay exceeds 0.2 μs, its further increase will cause a sharp BER rise. So, we should control the delay below 0.2 μs to keep the downlink demodulation effective. However, none of the prior synchronization schemes of backscatter systems can meet this requirement. For example, the synchronization error of MIXIQ is as high as 0.8 μs [18], and that of Saiyan [19] is even worse, which is of milliseconds. Hitchhike [8] does not fit either since its synchronization accuracy is around 2 μs. Although the most recent work, SyncScatter [15], can realize synchronization accuracy as low as 0.15 μs. It only works with excitations from a fixed transmitter and may degrade to over 1 μs synchronization error due to the dynamic power ramp time of ambient WiFi packets [22]. Actually, all rising edge-based synchronization, including [8], [15], [23], suffers from this drawback. As a result, our goal is to design a symbol synchronization at the tag with accuracy within 0.2 μs.

2) Low-Power Synchronization Using Correlation: Different from all prior synchronization at tags, we use correlation to realize accurate timing. Correlation-based timing is widely used in active WiFi and shows its high performance in many real-world applications [24], [25]. Our synchronization at the tag is designed as follows: We choose the binary envelope corresponding to the leading symbols in the preamble as the reference waveform (the *template* for correlation). A new sample is collected using the rectifier circuit and the comparator in the FPGA every clock cycle. After the newest samples are correlated with the template, we use correlation peaks to identify symbol boundaries. Specifically, we judge the symbol boundary based on whether the correlation peak exceeds a predefined threshold. As the duration of a standard 802.11b WiFi symbol is 1 μs, the following boundaries can be inferred.


(a) Sync errors at different template lengths.

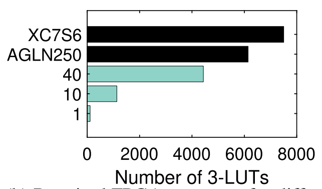
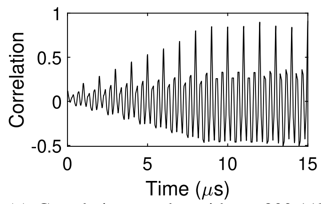

(b) Required FPGA resources for different templates.

Fig. 9. Optimal template length for low-power sync.


(a) Correlation results with an 802.11b WiFi packet.

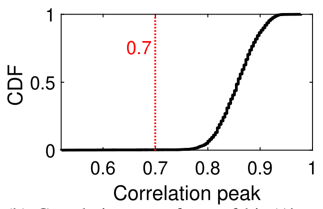

(b) Correlation waveform of bit '1' sequence with that of bit '0'.

Fig. 10. Correlation peaks to locate symbol boundaries.

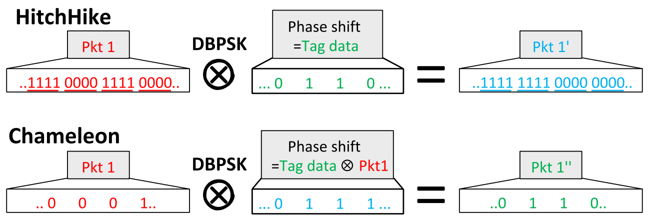


Fig. 11. Our tag considers both excitation and tag bits for backscatter modulation, making any random signal into a native WiFi packet. In contrast, codeword-translated-based systems, e.g., Hitchhike, are not native.

Correlation template design. Simply porting correlation-based synchronization for active WiFi would bring significant computation overhead for resource-constrained tags. However, if we oversimplify the correlation process, the synchronization error would be significantly increased. Hence, our design must carefully strike a balance between computation overhead and accuracy.

- First, we reduce computation overhead by making sample points binary instead of 10 or 16 bits for each sample in active WiFi.
- Second, we attempt to find a minimum template length that can deliver decent performance. In particular, we vary the template length and plot the results in Fig. 9a. If the template is as long as 1 symbol, the synchronization error drops below 0.1 μs only when the SNR exceeds 30dB, which is unacceptable. In contrast, when the template contains 10 or 40 symbols, the required SNRs are 20dB and 18dB, respectively. Hence, we pick 10 as a candidate for template length but need to verify its consumed hardware resources later.
- Third, we choose to use addition to replace multiplication for further computation saving. Multiplication of two 1-bit operands can be realized using an **AND** gate. The FPGA can sum up the output of all **AND** gates to obtain the correlation results.

As an example, we put the whole synchronization operation on an ultra-low-power nano-AGLN250 FPGA. This simplified correlation can be achieved using three-input look-up tables (3)-LUT). Based on this, we calculate how many 3-LUTs are necessary for different template lengths, and the results are shown in Fig. 9. When the template lengths are 1-, 10-, and 40-symbol long, the required numbers of 3-LUTs are 111, 1131, and 4435, respectively. As a comparison, the Microsemi AGLN250 FPGA can provide as many as 6144 ones, and the Xilinx XC7S6 FPGA has 7504 ones. Therefore, we confirm that setting the template length at 10 makes a good tradeoff between hardware resources and synchronization accuracy.

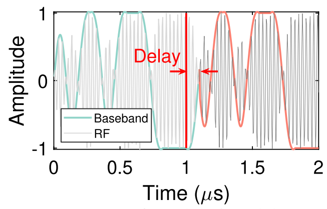
Synchronization accuracy. To verify the feasibility of our scheme, we test it with 1000 packets of random payload. We plot a representative correlated waveform in Fig. 10a. As each packet starts at 0 μs, we observe that the correlation has a high peak at 10 μs (corresponding to the end of the tenth symbol), which means that our tag is well synchronized to the tested WiFi packets. Furthermore, in over 99.9% cases, the peak is higher than 0.7, as shown in Fig. 10b. Knowing that the peak is above 0.7, we further make sure that this peak is higher than the neighboring peaks. Therefore, we can find the target symbol boundaries and then infer the following symbol locations.

Summary. Finally, we summarize the demodulation in the following steps. The tag counts the duration of the high envelope pulse and resets at every rising edge. At the same time, the tag monitors the correlation result. If a peak is detected, we demodulate the current bit by determining whether the pulse width is long enough.

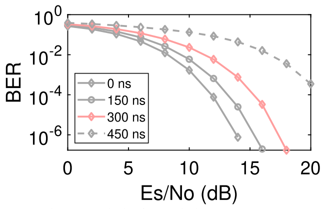
D. On-the-Fly Modulation

After we demodulate content-varying WiFi packets, our next step is to modulate those random packets into native WiFi packets. In particular, we present our modulation scheme and then demonstrate how we handle modulation delay and demodulation errors. Of course, the frequency shifting is leveraged to avoid interference from the carrier signal [5].

1) Modulation Scheme: Given tag data, e.g., {. . . , 0, 1, 1, 0, . . . }, and content-varying excitations, a native WiFi backscatter system should generate WiFi packets exactly containing only tag data in the payload. Such a design goal would enable a wide range of novel applications, as any standard WiFi device can directly demodulate this backscattered signal without any problems. However, none of the prior systems can achieve this because their modulation depends only on the tag data. From the above definition, we know there is no way to be native if the modulation is unaware of varying content because the target packet is fixed. As a result, we design on-the-fly modulation, which removes the uncertainty brought by the content-varying WiFi packets. In particular, it includes the demodulation results in our modulation, i.e., our modulation depends on carrier data and tag data at the same time. For example, as shown in Fig. 11, Hitchhike [8] applied phase shift modulation according to tag data only, making the requirements of excitation and backscattered data necessary for demodulating tag data. In contrast, our modulation uses the

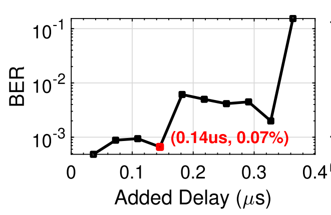


(a) Cause of modulation delay.

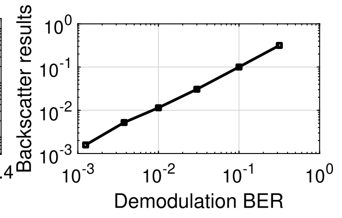


(b) Modulation delay vs BER.

Fig. 12. Modulation delay and its impact.



(a) Timing offset vs BER.



(b) Impact of demodulation BER and PER on backscatter modulation.

Fig. 13. On-the-fly modulation performance.

difference between tag data and carrier data, naturally leading to a native packet.

We first store the current tag data bit in a register. After the carrier data is demodulated, we immediately obtain the modulation bit as the **XOR** of the demodulated bit and the tag bit. Then, after applying the modulation bit to the carrier, the backscattered bit makes the perfect transformed tag bit as expected. In short, given content-varying excitations, our on-the-fly modulation, which includes dynamic demodulation results, can generate native target packets.

2) Modulation Delay: To ensure robust modulation, our on-the-fly modulation needs to take care of unavoidable modulation delays caused by symbol boundary tracking. After we locate a symbol boundary, a current symbol's start has been missed. Hence, the intended phase shift cannot be applied to the whole excitation symbol. As depicted in Fig. 12a, the phase shift occurs at the label $Delay$, which is after the start of a new symbol. To accurately quantify the impact of this delay, we perform another set of controlled experiments, where each case is tested with 1000 random packets with various delays. From the modulation BER results shown in Fig. 12b, when the delay is less than 0.3 μs, BERs rise slightly. When the delay is 0.45 μs, we have to increase 8dB for signals to realize similar BERs at a 0.3 μs delay. From these experiments, we can conclude that if the delay is within 0.3 μs, the backscatter quality is adequate for many real-world applications.

Next, we want to examine whether the empirical modulation delay can meet the above goal. In real-world scenarios, this modulation delay includes timing offset caused by the synchronization algorithm and hardware delay brought by the passive rectifier, the comparator, and FPGA processing. In our ADS simulation, the 40 MHz passive rectifier introduces a delay of about 0.01 μs, and the low-power comparator takes about 0.05 μs. As for the FPGA processing delay, half of the time corresponding to the first Barker chip of a symbol, whose specific duration is $\frac{1}{11}$ μs, should also be countered. That is because it is vital for demodulation, and only after demodulation will the tag be able to conduct on-the-fly modulation. In our implementation, the FPGA spends additional 3 clock cycles on logic processing and propagation. Therefore, the whole delay (excluding timing offset) can be estimated to: 0.01 μs (by rectifier) + 0.05 μs (by comparator) + $\frac{1}{2} \times \frac{1}{11}$ μs (by half chip for decoding) + $3 \times \frac{1}{55}$ $\mu s = 0.16$ μs. There is only $0.3 - 0.16 = 0.14$ (μs) for the synchronization algorithm, which can be easily satisfied as long as SNR exceeds 20dB, as shown in Fig. 9a.

Since it is nearly impossible to measure the accuracy of symbol synchronization in practice, here we investigate its impact indirectly through end-to-end experiments. Besides the delays introduced by the synchronization circuit or FPGA processing, we manually delay the backscatter modulation at the tag and then observe BERs using a COTS WiFi NIC. Results are shown in Fig. 13a. When the additional delay is within 0.14 μs, the decoding BER of backscatter modulation is below 0.1%. The BER is not as low as in the simulation results of Fig. 12b. That is because, in real-world experiments, ambient WiFi transmissions interfere with the backscatter modulation. On the contrary, when the delay exceeds 0.14 μs, the BER increases significantly. These results show that our synchronization design provides a margin of around 0.14 μs, which meets our design goal.

3) Impact of Demodulation Errors at Tags: Different from prior works, our backscatter modulation is highly dependent on the demodulation quality at tags. If the demodulation results are incorrect, backscatter transmissions will be significantly affected. This is a limitation of Chameleon. Hence, we need to investigate how demodulation BERs at tags affect the backscatter transmission performance.

As 1 Mbps WiFi signals are modulated using DBPSK, the incorrect bits bring additional phase shifts to all the following backscatter modulated symbols. However, the phase differences between them, which are used in tag data decoding at the WiFi receiver, are kept intact. That means demodulation errors at a tag will cause about the same number of backscatter errors. One may also be concerned that the demodulation errors in *preamble* or *header* will destroy backscatter transmission as they carry necessary information such as the packet modulation method and the packet size. However, this problem is not significant. First, the preamble contains 144 fixed bits. Chameleon tags can store them and then correct the corresponding errors. Second, the header contains only 32-bit packet information that is protected using the CRC. The demodulation of such a small field is much easier. Even with errors, they can be detected using the CRC. Thus, we manually introduce different amounts of demodulation errors in the *PSDU* field of the backscattered packets in controlled experiments and then observe the backscatter modulation BER. As shown in Fig. 13, the backscatter BER is very close to the demodulation BER at the tag. That means our backscatter modulation can tolerate a decent amount of demodulation errors at tags.

E. Clock Error Compensation

Besides time synchronization, clock frequency error is another concern for backscatter systems. In WiFi PHY, there

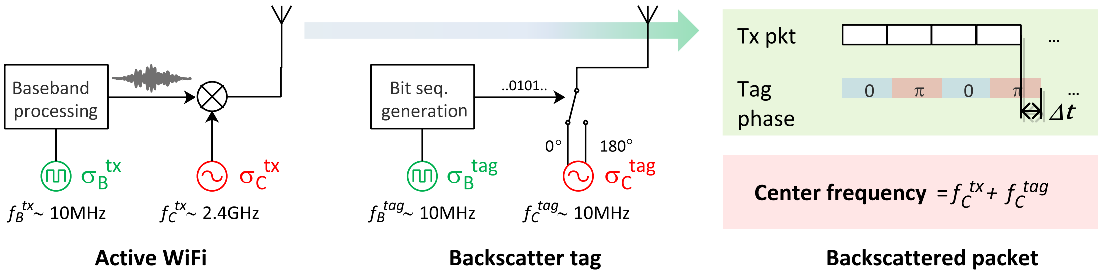


Fig. 14. Clocks and their errors in backscatter systems.

are two clock signals: the baseband clock frequency f_B^{tx} and the carrier frequency f_C^{tx}. They are shown in Fig. 14. f_B^{tx} is used for baseband signal processing, and it is typically tens of megahertz. The carrier frequency f_C^{tx} for shifting the baseband signal to the radio band is around 2.4 GHz. According to the WiFi standard, the relative errors of those two clocks should both be better than $\pm 25ppm$ [26] (ppm: $parts\ per\ million, 10^{-6}$). Corresponding to f_B^{tx} and f_C^{tx}, backscatter tag also has two clock frequencies: the baseband computation and control frequency f_B^{tag} and the frequency-shift clock frequency f_C^{tag}, respectively. Tag reuses RF signals generated by active WiFi devices instead of itself. Thus, these clock frequencies are both tens of megahertz. Intuitively, as the backscatter tag bridges the transmitter and the receiver, it may encounter stricter clock requirements. An intuitive thought is that the total tolerance ($\pm 25ppm$) minus the exciter clock error is the tag clock error tolerance:

$$\begin{array}{ccc} Total\ tolerance- & TX\ error & = & Tag\ tolerance \\ (i.e., \pm 25ppm)- & (i.e., \pm 20ppm) & = & (\pm 5ppm) \end{array}$$

If this is true, there is little space left for clock signals in backscatter tags. After further analysis, we find it is not that direct. What really matters is the frequency error in backscattered packets, which should not exceed $\pm 25ppm$. It is influenced by clock signals in both the active WiFi transmitter and the backscatter tag. We consider them as a whole and analyze the baseband frequency f_B^{tx} and carrier frequency f_C^{tx}.

Baseband clock. The influence of frequency error in the baseband clock mainly focuses on the gradually increased misalignment between backscatter modulation and excitation symbols, which is visually shown in the right side of Fig. 14. For this reason, the DSSS chips in a symbol may be modulated with different tag bits, which leads to communication errors. As defined, let f_B be the standard baseband processing clock, then the relationship in Equation (1) is satisfied. Let t, t_{tx}, and t_{tag} be the standard time, the time in WiFi, and the time at the tag, respectively. Digital systems infer time by counting the baseband clock cycles. The relationship between these times is shown in Equation (2). Thus, the misalignment of backscatter modulation in Fig. 14 can be inferred as Equation (3).

$$\begin{aligned} f_B^{tx} &= f_B \times (1 + \sigma_B^{tx}) \\ f_B^{tag} &= f_B \times (1 + \sigma_B^{tag}) \end{aligned} \tag{1}$$

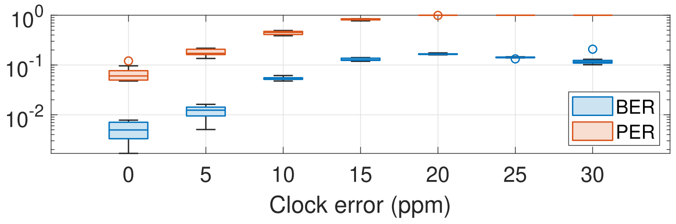


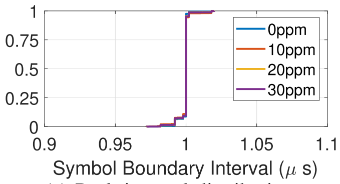
Fig. 15. End-to-end performance over σ_B^{tag}.

$$\begin{aligned} count &= t \times f_B \\ &= t_{tx} \times f_B^{tx} \\ &= t_{tag} \times f_B^{tag} \end{aligned} \tag{2}$$

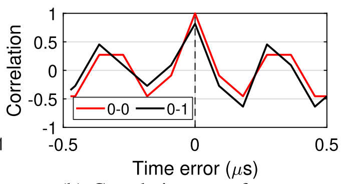
$$\begin{aligned} \Delta t &= t_{tag} - t_{tx} \\ &= t \times (\sigma_B^{tx} - \sigma_B^{tag}) \end{aligned} \tag{3}$$

It can be observed that the timing error Δt is proportional to the difference of two clock errors: $\sigma_B^{tx} - \sigma_B^{tag}$. Therefore, to minimize the time misalignment, σ_B^{tag} should be kept as close to σ_B^{tx} as possible. The backscatter baseband processing clock may have a stricter tolerance than the active WiFi signals. To test the allowable tag baseband clock error, we conduct controlled experiments. An arbitrary waveform generator with clock error $\pm 1ppm$ is used to generate a clock signal with controllable errors, which then works as f_B^{tag}. The frequency shift clock of a tag relies on a precise crystal oscillator. The TI CC3200 WiFi module is used to generate WiFi packets as excitation. We test the end-to-end performance under different relative errors. Results are shown in Fig. 15. BER and PER rise significantly with an increased clock error. Only when the clock error is within 5ppm will PER be below 10%. Compared with the clock error tolerance of 25ppm in active WiFi, this is too strict for backscatter tags. Still, backscatter tags are expected to work well using an on-chip oscillator for consideration of power consumption and price. However, the on-chip oscillators based on CMOS circuits are significantly influenced by the temperature and voltage supply. Their typical errors are in the range of $\pm 1000ppm$ [27], [28], [29].

How can the Chameleon tag fit such on-chip clocks? As shown in Fig. 10a, we have observed that the correlation peaks in the backscatter tag appear periodically. Specifically, experiments show that the time intervals between adjacent peaks mostly fall into 1 μs, which exactly matches the time duration of a WiFi symbol. As shown in Fig. 16a, over 99.9% samples are within the range of 0.99 $\sim$ 1.01 μs. Moreover,



(a) Peak interval distribution.



(b) Correlation waveform.

Fig. 16. Correlation peak follows excitation clock.

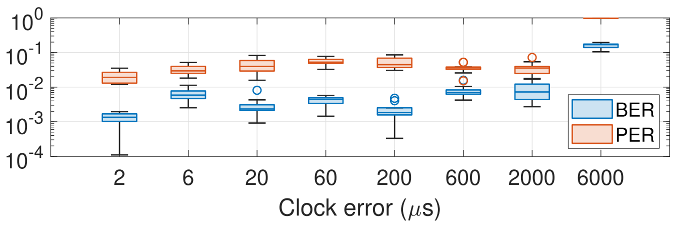


Fig. 17. End-to-end performance over σ_B^{tag} with compensation.

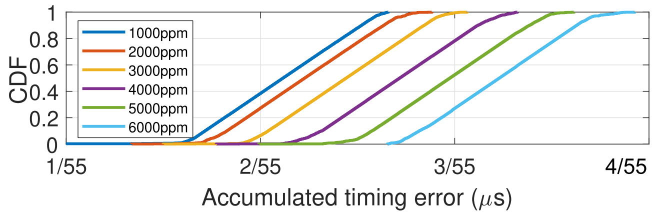


Fig. 18. Timing error over σ_B^{tag} with compensation.

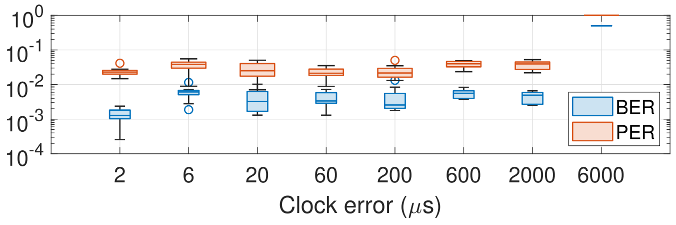
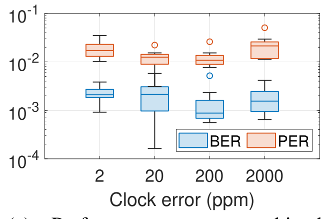
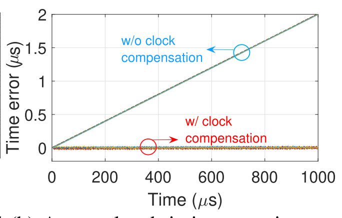


Fig. 19. End-to-end performance over σ_C^{tag}.



(a) Performance over combined clock.



(b) Accumulated timing error in tag.

Fig. 20. Tag works well with 2000ppm clock error.

even with clock errors, the peak intervals are still around 1 μs and share similar distributions to those when there is no clock error. One may be curious how a fixed template can be used to locate boundaries for both bits '1' and '0' in the presence of a random payload. This is because the envelope waveforms for bits '1' and '0' are quite similar. Their correlation peak appears when they are aligned in time, which can be seen in Fig. 16. Thus, the correlation peaks with an interval of 1 μs can serve as a reference clock for the backscatter tag. Once the tag locks to it, the time difference Δt in Equation (3) will be canceled out. Then, the tag has much better clock error tolerance.

The operation of locking to the excitation at a tag is very easy: calculating the correlation and finding the peak every 1 μs, then using it as the reference time in differential demodulation and on-the-fly modulation. Experiment results in Fig. 17 show that as long as the baseband clock error is better than $\pm 2000ppm$, the tag can realize good end-to-end communication performance. The BER is within 1% and the PER is within 10%. However, when the clock error is $\pm 6000ppm$, the performance degrades significantly. This is because when the clock error is too large, the tag will fail to lock to the excitation clock. To clarify this, we show the cumulative distribution function (CDF) of accumulated timing error with σ_B^{tag} ranging from 1000 ppm to 6000ppm in Fig. 18. The accumulated timing error under one clock error is distributed uniformly in a range around $1/55\mu s$, and its center increases with clock error. According to the results in Fig. 7a, the pulse widths for bits '0' and '1' cluster around $10/55\mu s$ and $15/55\mu s$, respectively, with a minimum separation of $3/55\mu s$. When the clock error is greater than 3000ppm, the timing error exceeds $3/55\mu s$. This will cause a demodulation error. Thus, the tolerable baseband clock error is $\pm 3000ppm$.

Frequency shift clock. A tag needs to shift the backscattered packet into another wireless frequency channel to avoid interference from excitation signals. Errors in this clock also influence backscatter communication. The center frequency of the target channel is $f_C^{tx} + f_C^{tag}$, and the error frequency is as Equation (4). The center frequency error tolerance is $\pm 25ppm$, which is expressed as Equation (5). Combining these two equations, we can get the toleratable range of σ_C^{tag} as Equation (6).

$$\Delta f = f_C^{tx} \times \sigma_C^{tx} + f_C^{tag} \times \sigma_C^{tag} \tag{4}$$

$$\left| \frac{\Delta f}{f_C^{tx} + f_C^{tag}} \right| < 25ppm \tag{5}$$

$$-\frac{f_C^{tx}}{f_C^{tag}} \times (25ppm + \sigma_C^{tx}) \leq \sigma_C^{tag} \leq \frac{f_C^{tx}}{f_C^{tag}} \times (25ppm - \sigma_C^{tx}) \tag{6}$$

In most cases, WiFi devices have a clock error margin, e.g., $|\sigma_C^{tx}| \leq 10ppm$. Combining this with the fact that $\frac{f_C^{tx}}{f_C^{tag}} \simeq 120$, the tag frequency-shifting clock tolerance can be as much as $\pm 1800ppm$. We test it in real-world experiments. The excitation device is a TI CC3200 WiFi module. As shown in Fig. 19, when the clock error is within $\pm 2000ppm$, its influences on BER and PER are negligible.

To sum up, the baseband clock and the carrier clock have a similar clock error tolerance around $\pm 2000ppm$ and similar frequency (22 MHz and 20 MHz, respectively), so we can use one clock source for both of them for power saving. When using a signal with errors ranging from 2ppm to 2000ppm as the source for both the baseband processing clock and the frequency-shit clock, respectively, the end-to-end communication performance is shown in Fig. 20a. When both clock errors are below 2000ppm, the PER is still well below 10%.

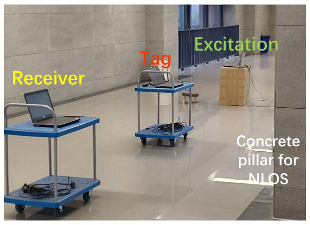


Fig. 21. Experiment scenario.

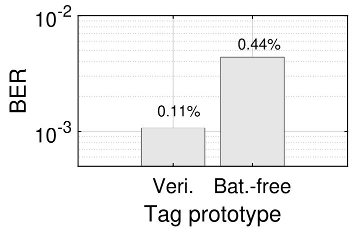


Fig. 22. Performance of two prototypes.

To show the effectiveness of the clock compensation design, we also test timing errors at the tag with 2000ppm clock errors. As shown in Equation (3), the timing error is proportional to both time and clock error. We capture the timing signals and analyze the errors. As can be seen in Fig. 20, without clock compensation, the timing error accumulates over time. When the packet lasts for 1000 μs, corresponding to a PSDU size of around 100 bytes, the timing error at the packet tail will be 2 μs, equivalent to the duration of two WiFi symbols. Meanwhile, when a tag locks to the excitation, the timing error can be kept very low.

III. Implementation

We build multiple prototypes to verify Chameleon's superiority in transmission performance and power consumption. The testing experiments are conducted in a 15 m × 30 m public area as shown in Fig. 21.

Verification prototype. We first build a verification prototype to explore the best performance of Chameleon. The key components, namely demodulation and backscatter modulation algorithms, are realized in a Microsemi AGLN250 nano FPGA. The tag modulation is realized using an ADI ADG902 RF-switch. In the decoding circuit, a passive rectifier consisting of capacitors and HSMS-2862 diodes, and a high-speed NCS2250 comparator are used. As shown in Fig. 23a, our verification prototype includes two 15dB SKY65405-21 low noise amplifiers (LNA), which are designed for better downlink range. In this prototype, synchronization and backscatter modulation algorithms are run at 55 MHz. Note that a DSSS chip lasts for $\frac{1}{11}$ μs, which can be easily recognized with a 20 MHz clock. In other words, the synchronization requirements are satisfied.

Battery-free prototype. We also design a battery-free prototype, as shown in Fig. 23b. While some parts are reused from the verification prototype, LNA is not included for power reduction. We design a circuit to wake up the FPGA from the *Flash*Freeze* state when excitation signal arrives. It is composed of a low-bandwidth passive rectifier and a

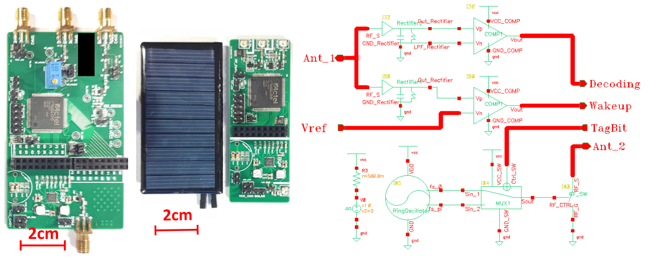


Fig. 23. Tag-1: verification prototype; Tag-2: battery-free prototype; Tag-3: IC simulation.

low-power comparator, the NCS2200. The system clock rate is also reduced from 55 MHz to 22 MHz. The direct benefit is a power reduction of about 2.5× for digital processing in FPGA. Moreover, this also leads to a shorter reference sequence for correlation-based synchronization, which also lowers the system power consumption. For RF energy harvesting, a similar rectifier is used to convert RF signals to DC. Then, a TI BQ25570 harvesting management chip stores energy in a capacitor and intermittently powers the system up.

It may be concerned about whether the battery-free prototype can do the job. We test the end-to-end performances of the verification prototype and the battery-free prototype with identical settings. Results are shown in Fig. 22. BER of 0.44% is still acceptable for many applications.

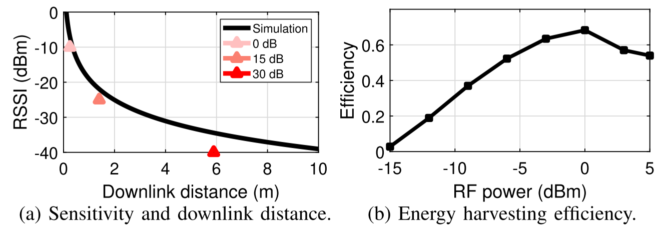
Power consumption comparison. We use a Keysight 34450A to measure power consumption for the above prototypes. Results are shown in Table I. The verification prototype costs 90.19 mW in total, including 15.41 mW by FPGA, 590 μW by an oscillator, 1.83 mW by RF-switch, and 72.36 mW by the wakeup and decoding circuits. The main dissipation comes from the FPGA, which runs at a high frequency. The wake and demodulation circuits contain two LNAs, which take about 72 mW and are not included in the battery-free prototype. Compared with the verification prototype, the power consumed by the FPGA and that by wakeup/demodulation circuits are reduced to 5.23 mW and 360 μW in the battery-free design, respectively. In summary, over 10× power reduction is realized in the battery-free prototype.

IC prototype. Although the battery-free prototype consumes much less power than what the verification prototype does, it is possible to make further power savings using ASIC techniques. In particular, we simulate an IC prototype using Cadence IC6.17 Virtuoso software and TSMC 0.18 μm CMOS process design kits. A ring oscillator made up of cascaded comparators instead of PLL in FPGA as shown in Fig. 3 is used to generate clock signals. The total power is estimated to be about 595.8 μW, which is only 0.66% of the verification prototype.

Ambient excitations and COTS receivers. We use a TI CC3200 Wireless MCU (Campus WiFi) to provide excitation signals. For end-to-end and micro-benchmark experiments, we primarily deploy a laptop equipped with a Qualcomm Atheros AR938X NIC as the backscatter receiver because our competitor, e.g., FS-Backscatter, requires a WiFi NIC that can turn off CRC check. For the receivers in our application

TABLE I
POWER CONSUMPTION OF THREE PROTOTYPES

	power consumption breakdown				Total
	Digital core	Oscillator	RF-switch	Wakeup+Demod.	
Verification	15.41 mW (100%)	590 μW (100%)	1.83 mW (100%)	72.36 mW (100%)	90.19 mW (100%)
Battery-free	5.23 mW (36.53%)	590 μW (100%)	1.83 mW (100%)	360 μW (0.50%)	8.41 mW (9.32%)
IC	324 μW (2.10%)	199.7 μW (33.84%)	2.41 μW (0.13%)	67.58 μW (0.09%)	595.8 μW (0.66%)



(a) Sensitivity and downlink distance. (b) Energy harvesting efficiency.

Fig. 24. Backscatter sensitivity and energy harvesting performance.

experiments, we include a range of COTS WiFi devices, including laptops, iOS devices, and Android devices, to examine Chameleon's compatibility.

Sensitivity and downlink range. As the rectifier bandwidth in the demodulation circuit is as high as 40 MHz, the corresponding sensitivity is estimated at about -10 dBm. Weaker excitation may still have enough quality to realize the downlink decoding, but it is not high enough to drive the comparator. We set the transmitter power to be 18 dBm, which is allowed for WiFi devices in most areas [30]. The received signal strength indicator (RSSI) with the free space loss model [31] at various distances is depicted in Fig. 24a. The -10 dBm sensitivity corresponds to a distance below 0.5 m, which is too short. Meanwhile, the RSSI below -85 dBm in the traditional 802.11b WiFi is enough to provide a high SNR for demodulation [26]. We can use this as a reference to infer the achievable sensitivity of Chameleon in theory. It uses only one Barker chip of all eleven (causing 11× SNR reduction) and focuses on its envelope amplitude like binary amplitude shift keying (2ASK) (causing 2× SNR reduction compared with BPSK [21]). Thus, in terms of SNR, it is possible to boost the sensitivity to $-85\ dBm \times 11 \times 2 = -71\ dBm$ using LNA, as done in related works [15], [18], [19]. However, LNA with a higher gain causes higher power consumption. A tradeoff between power consumption and sensitivity is needed. 15dB and 30dB LNAs improve effective downlink ranges to 1.4 m and 5.9 m, respectively. We believe that the latter is appropriate for most indoor IoT applications and choose it for our system.

Energy Harvesting efficiency. We also examine the energy harvesting efficiency of our battery-free tag prototype. In particular, we record the needed time T to charge the capacitor C_{store} from empty to full, in which procedure the capacitor voltage is increased from V_{min} to V_{max}. Combining the input RF power P_{in}, the energy harvesting efficiency can be calculated as: $\eta = \frac{C_{store}(V_{max}^2 - V_{min}^2)}{2P_{in}T}$. The results with various RF power levels are shown in Fig. 24. The energy harvesting efficiency grows from 2.8% to 68.27% when the power level of RF signals increases from -15 dBm to 0 dBm. That is because with weak RF signals, the dissipation caused by the diode forward voltage in the rectifier and charging management chip BQ25570 becomes significant, resulting in low energy harvesting efficiency. When RF power is below -15 dBm, such dissipation becomes unaffordable, so energy harvesting fails.

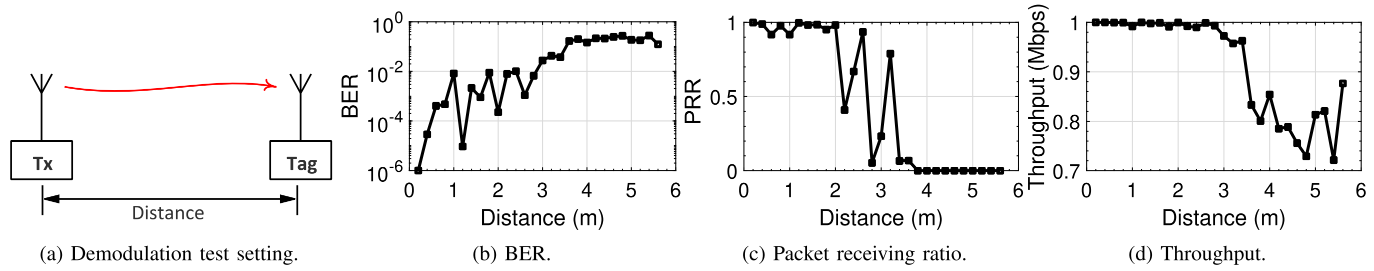
IV. PERFORMANCE EVALUATION

We first evaluate demodulation and modulation quality. After that, we investigate end-to-end performance and compare Chameleon with prior works.

A. Micro Benchmarks

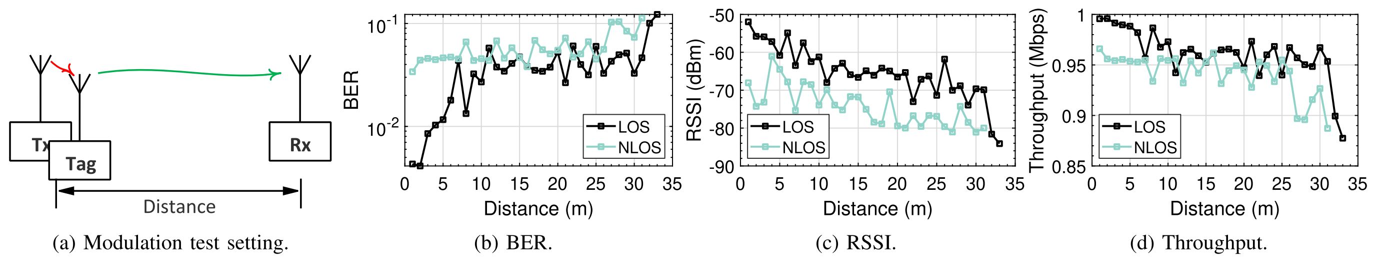
Differential demodulation. We conduct tests with different Tx-tag distances and set the transmitter to generate packets at the power level of 18 dBm. For each position, over 1000 random packets are sampled. The results are shown in Fig. 25. We observe that when the downlink range is below 1 m, the BER is less than 0.1%. When the range increases to 3 m, the BER jumps to 1%. Meanwhile, the packet receiving ratio (PRR) is above 90% when the downlink range is within 2 m. Moreover, when the downlink range is shorter than 3 m, the throughput is always above 0.95 Mbps. Finally, the throughputs in all tested ranges are above 0.7 Mbps. All results indicate that our tag is able to demodulate WiFi signals reliably.

On-the-fly modulation. Next, we examine how on-the-fly modulation performs. Since the results of the synchronization algorithm have been shown in the previous section, we focus on its performance with different tag-Rx ranges. As shown in Fig. 26, a tag is placed 0.3 m away from the WiFi transmitter. Note that the previous discussion shows that the downlink demodulation can achieve nearly perfect excitation bits with this range. For NLoS experiments, we use a load-bearing concrete pillar to block signals from the transmitter to the tag. From the results in Fig. 26, we see that BERs rise with increasing distances for both LoS and NLoS scenarios. In LoS cases, the BER is below 1% when the distance is shorter than 4 m. While in NLoS settings, the BER is above 3% at a distance of 1 m, which is mainly due to the decreased RSSIs at longer distances and NLoS blockage. In addition, we observe that our tag supports maximum uplink ranges of 33 m and 30 m for LoS and NLoS scenarios, respectively. Furthermore, the backscatter throughput is above 0.85 Mbps in all tested settings. Such performance is attributed to our on-the-fly modulation, which can tolerate reasonable delays and demodulation errors. Experimental results in the *Section II* of *Supplementary Material* show that when differential demodulation and on-the-fly modulation are combined, Chameleon still works well in a certain area.



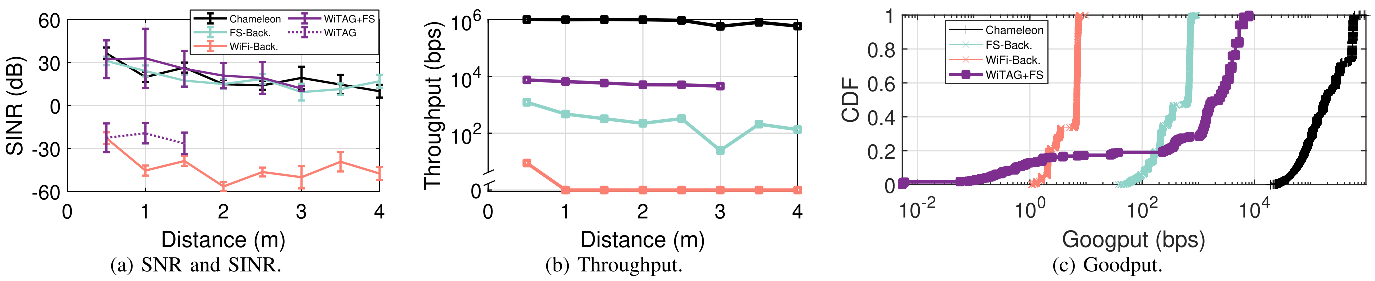
(a) Demodulation test setting. (b) BER. (c) Packet receiving ratio. (d) Throughput.

Fig. 25. Differential demodulation performance.



(a) Modulation test setting. (b) BER. (c) RSSI. (d) Throughput.

Fig. 26. On-the-fly modulation performance.



(a) SNR and SINR. (b) Throughput. (c) Goodput.

Fig. 27. Comparison with the state-of-the-art systems.

B. End-to-End Performance

Comparisons with prior works. Even though none of the prior systems support native WiFi backscatter with random excitations, we make a comparatively loose selection criterion for our competitors: supporting content-varying excitations and a single COTS receiver at the same time. Following this criterion, TScatter [14], BackFi [4], and CAB [16] are not included because they do not support COTS WiFi receivers. Hitchhike [8], MOXcatter [10], and SubScatter [17] are not good candidates since they use two COTS receivers. Interscatter [6] and Passive WiFi [7] can indeed generate native WiFi packets, but they only work with CW excitations. Finally, we decide to compare our Chameleon with three closest works: WiTAG [11], FS-Backscatter [5], and WiFi Backscatter [1]. For WiTAG, we use the Xilinx Zynq-7000 + AD9361 FMCOMMS3 platform to generate A-MPDU 802.11n packets. We set the HT-MCS to 7 and use a power amplifier to boost the signal power to about 18 dBm. In the experiment, 1000 random WiFi packets are generated every second.

We first compare the signal to the interference plus noise ratio (SINR) at the receiver side. WiFi backscatter and WiTAG do not frequency-shift backscattered packets into another channel to reduce self-interference. As a result, their SINRs are around -30dB, which can be shown in Fig. 27a. This results in poor communication performance. To investigate the benefit from the usage of A-MPDU in backscatter, we compare our Chameleon with WiTAG + frequency-shift instead (denoted as "WiTAG+FS" in Fig. 27a). Chameleon, WiTAG+FS, and FS-Backscatter avoid interference from excitation and realize good SINR thanks to frequency shift. Results in Fig. 27b show that Chameleon realizes a throughput of 992.4 kbps, while those for WiTAG+FS and FS-Backscatter are 8 kbps and 1 kbps, respectively. The WiFi Backscatter only achieves a throughput of 9 bps at 0.5 m and barely works with longer ranges. For WiTAG+FS and FS-Backscatter, lost packets and incorrectly received packets caused by ambient interference introduce wrong bits. This causes a significant performance drop with increasing ranges. For example, the throughput of FS-Backscatter is lower than 100 bps when the distance is over 3 meters. The root cause of the above differences is that WiTAG+FS, FS Backscatter, and WiFi Backscatter all conduct packet-level modulation. In contrast, Chameleon achieves symbol-level backscatter and supports a much higher throughput.

Furthermore, we compare the goodputs of WiTAG+FS, FS Backscatter, and WiFi-Backscatter. The excitation rate is set at 1000 packets per second. Note that goodput is the effective data rate at the application layer. Packets with incorrect CRCs cannot go through the MAC layer and make no contribution to the goodput. Chameleon realizes a goodput of about 0.28 Mbps, while those for WiTAG+FS, FS-Backscatter, and WiFi-Backscatter are 2.51 kbps, 0.47 kbps, and 5.46 bps, respectively. Notice that these rates are all below their theoretical maximum values. This is possibly due to the high packet failure at high rates and intensive ambient interference. It can also be seen that WiTAG+FS goodput varies from 0.01 bps to 8 kbps, and it is below 1 bps in about 13% cases. That is mainly because WiTAG conveys bits '0' and '1' using the correct and incorrect packets, respectively. This means the PER of carrier packets can be seen as WiTAG BER. Ambient interference makes such transmissions very fragile.

We also observe that Chameleon is compatible with WiFi on multiple kinds of commercial devices, including WiFi routers, Android devices, Windows devices, and Apple devices. Specific methods and results can be seen in *Section IV* of *Supplementary Material*.

V. Related Work

Backscatter has been a hot topic in the wireless and networking community [32], [33], [34], [35], [36], [37], [38], [39], [40] and Chameleon makes contribution at the following two forefronts.

Commodity backscatter. There are a great deal of prior works that study how to design backscatter systems using commodity radios as receivers, including WiFi backscatter [8], [9], [15], Bluetooth backscatter [41], [42], [43], LTE backscatter [23], ZigBee backscatter [9], and LoRa backscatter [44], [45], [46], [47]. They all eliminate the need for dedicated hardware as exciters and receivers but fail to achieve native commodity connectivity. This is because they adopt codeword translation alike techniques to deal with content-varying ambient excitations, and demand two receivers. To break this barrier, a number of single-receiver backscatter systems are proposed [16], [48], [49], [50], [51]. Unfortunately, these works either pose restrictions on excitations or require SDRs for tag data demodulation. Unlike these works, Chameleon proposes a novel on-the-fly modulation and enables native WiFi backscatter for the first time.

Ambient signal demodulation. Another key function of backscatter is to demodulate signals from ambient environments. Past works design various demodulations at the packet level [8], [9], [10] and symbol level [18], [19], [20]. While these approaches achieve either better throughput [19] or longer ranges, e.g., a downlink range of 52 m [20], none of them can demodulate ambient WiFi signals. In contrast, Chameleon builds the first passive WiFi demodulator by turning rectifying phase-modulation signals into ASK signals.

VI. Conclusion

In this paper, We have presented Chameleon, the first native WiFi backscatter system with uncontrolled content-varying excitations. In this system, the low-power tag can demodulate WiFi signals in an ASK way without the need to differentiate the symbol phases. On top of that, we have designed on-the-fly modulation that can turn any excitations into a native WiFi packet. Extensive comparisons and experiments with COTS devices have demonstrated Chameleon's superior performance and strong compatibility with different kinds of WiFi devices. We envision that our Chameleon design will open the door to making battery-free WiFi connectivity ubiquitous for real-world IoT applications.

References

[1] B. Kellogg, A. Parks, S. Gollakota, J. R. Smith, and D. Wetherall, "Wi-Fi backscatter: Internet connectivity for RF-powered devices," in *Proc. ACM Conf. SIGCOMM*, Aug. 2014, pp. 607–618.

[2] *Global Wi-Fi Enabled Devices Shipment Forecast*. Accessed: Jul. 2020. [Online]. Available: https://www.researchandmarkets.com/reports/5135535/global-wi-fi-enabled-devices-shipment-forecast

[3] N. Gershenfeld, R. Krikorian, and D. Cohen, "The Internet of Things," *Sci. Amer.*, vol. 291, no. 4, pp. 76–81, 2004.

[4] D. Bharadia, K. R. Joshi, M. Kotaru, and S. Katti, "BackFi: High throughput WiFi backscatter," in *Proc. ACM Conf. Special Interest Group Data Commun.*, Aug. 2015, pp. 283–296.

[5] P. Zhang, M. Rostami, P. Hu, and D. Ganesan, "Enabling practical backscatter communication for on-body sensors," in *Proc. ACM SIGCOMM Conf.*, Aug. 2016, pp. 370–383.

[6] V. Iyer, V. Talla, B. Kellogg, S. Gollakota, and J. Smith, "Inter-technology backscatter: Towards internet connectivity for implanted devices," in *Proc. ACM SIGCOMM Conf.*, Aug. 2016, pp. 356–369.

[7] B. Kellogg, V. Talla, S. Gollakota, and J. R. Smith, "Passive Wi-Fi: Bringing low power to Wi-Fi transmissions," in *Proc. USENIX NSDI*, 2016, pp. 151–164.

[8] P. Zhang, D. Bharadia, K. Joshi, and S. Katti, "HitchHike: Practical backscatter using commodity WiFi," in *Proc. ACM SenSys*, 2016, pp. 259–271.

[9] P. Zhang, C. Josephson, D. Bharadia, and S. Katti, "FreeRider: Backscatter communication using commodity radios," in *Proc. ACM CONEXT*, 2017, pp. 389–401.

[10] J. Zhao, W. Gong, and J. Liu, "Spatial stream backscatter using commodity WiFi," in *Proc. ACM MobiSys*, 2018, pp. 191–203.

[11] A. Abedi, F. Dehbashi, M. H. Mazaheri, O. Abari, and T. Brecht, "WiTAG: Seamless WiFi backscatter communication," in *Proc. ACM SIGCOMM*, 2020, pp. 240–252.

[12] J. Zhao, W. Gong, and J. Liu, "Towards scalable backscatter sensor mesh with decodable relay and distributed excitation," in *Proc. ACM MobiSys*, 2020, pp. 67–79.

[13] W. Gong, L. Yuan, Q. Wang, and J. Zhao, "Multiprotocol backscatter for personal IoT sensors," in *Proc. ACM CoNEXT*, 2020, pp. 261–273.

[14] X. Liu, Z. Chi, W. Wang, Y. Yao, P. Hao, and T. Zhu, "Verification and redesign of OFDM backscatter," in *Proc. USENIX NSDI*, 2021, pp. 939–953.

[15] M. Dunna, M. Meng, P. Wang, C. Zhang, P. P. Mercier, and D. Bharadia, "SyncScatter: Enabling WiFi like synchronization and range for WiFi backscatter Communication," in *Proc. USENIX NSDI*, 2021, pp. 923–937.

[16] Y. Yang, L. Yuan, J. Zhao, and W. Gong, "Content-agnostic backscatter from thin air," in *Proc. 20th Annu. Int. Conf. Mobile Syst., Appl. Services*, Jun. 2022, pp. 343–356.

[17] L. Yuan and W. Gong, "SubScatter: Sub-symbol WiFi backscatter for high throughput," in *Proc. IEEE 30th Int. Conf. Netw. Protocols (ICNP)*, Oct. 2022, pp. 1–11.

[18] M. Rostami, X. Chen, Y. Feng, K. Sundaresan, and D. Ganesan, "MIXIQ: Re-thinking ultra-low power receiver design for next-generation on-body applications," in *Proc. 27th Annu. Int. Conf. Mobile Comput. Netw.*, Oct. 2021, pp. 364–377.

[19] X. Guo et al., "*Saiyan*: Design and implementation of a low-power demodulator for LoRa backscatter systems," in *Proc. USENIX NSDI*, 2022, pp. 437–451.

[20] S. Li et al., "Passive DSSS: Empowering the downlink communication for backscatter systems," in *Proc. USENIX NSDI*, 2022, pp. 913–928.

[21] F. Xiong, *Digital Modulation Techniques*. Artech House, 2006.

[22] Z. Huang and W. Gong, "EAScatter: Excitor-aware Bluetooth backscatter," in *Proc. IEEE/ACM 30th Int. Symp. Quality Service (IWQoS)*, Jun. 2022, pp. 1–10.

[23] Z. Chi, X. Liu, W. Wang, Y. Yao, and T. Zhu, "Leveraging ambient LTE traffic for ubiquitous passive communication," in *Proc. ACM SIGCOMM*, 2020, pp. 172–185.

[24] *A GNURadio-Based Implementation of WiFi Direct Sequence Spread Spectrum Communications*. Accessed: Jul. 2023. [Online]. Available: https://github.com/hui811116/gr-wifi-dsss

[25] *Keysight Advanced Design System (ADS)*. Accessed: Jul. 2023. [Online]. Available: https://www.keysight.com/us/en/products/software/pathwave-design-software

[26] *IEEE Standard for Information Technology—Telecommunications and Information Exchange Between Systems—Local and Metropolitan Area Networks—Specific Requirements—Part 11: Wireless Lan Medium Access Control (MAC) and Physical Layer (PHY) Specifications—Redline*, IEEE Standard 802.11-2007, 2007.

[27] K. Sundaresan, P. E. Allen, and F. Ayazi, "Process and temperature compensation in a 7-MHz CMOS clock oscillator," *IEEE J. Solid-State Circuits*, vol. 41, no. 2, pp. 433–442, Feb. 2006.

[28] E. J. Pankratz and E. Sanchez-Sinencio, "Multiloop high-power-supply-rejection quadrature ring oscillator," *IEEE J. Solid-State Circuits*, vol. 47, no. 9, pp. 2033–2048, Sep. 2012.

[29] M. Choi, T. Jang, S. Bang, Y. Shi, D. Blaauw, and D. Sylvester, "A 110 nW resistive frequency locked on-chip oscillator with 34.3 ppm/°C temperature stability for system-on-chip designs," *IEEE J. Solid-State Circuits*, vol. 51, no. 9, pp. 2106–2118, Sep. 2016.

[30] *Maximum WiFi Transmission Power Per Country*. Accessed: Jul. 2023. [Online]. Available: https://w.wol.ph/2015/08/28/maximum-wifi-transmission-power-country/

[31] *Free-space Path Loss*. Accessed: Jul. 2023. [Online]. Available: https://en.wikipedia.org/wiki/Free-space_path_loss

[32] M. K. Mukerjee, D. Naylor, J. Jiang, D. Han, S. Seshan, and H. Zhang, "Practical, real-time centralized control for CDN-based live video delivery," in *Proc. ACM Conf. Special Interest Group Data Commun.*, Aug. 2015, pp. 311–324.

[33] R. Ghaffarivardavagh, S. S. Afzal, O. Rodriguez, and F. Adib, "Ultra-wideband underwater backscatter via piezoelectric metamaterials," in *Proc. ACM SIGCOMM*, 2020.

[34] L. Chen, W. Hu, K. Jamieson, X. Chen, D. Fang, and J. Gummeson, "Pushing the physical limits of IoT devices with programmable meta-surfaces," in *Proc. USENIX NSDI*, 2021, pp. 425–438.

[35] Q. Dong, J. Wu, W. Hu, and J. Crowcroft, "Practical network coding in wireless networks," in *Proc. ACM MobiCom*, 2007, pp. 306–309.

[36] S. Jog et al., "Enabling IoT self-localization using ambient 5G signals," in *Proc. USENIX NSDI*, 2022, pp. 1011–1026.

[37] Y. Zhao et al., "Towards battery-free machine learning and inference in underwater environments," in *Proc. ACM HotMobile*, 2022, pp. 29–34.

[38] J. Jang and F. Adib, "Underwater backscatter networking," in *Proc. ACM SIGCOMM*, 2019, pp. 187–199.

[39] W. Gong, S. Chen, J. Liu, and Z. Wang, "MobiRate: Mobility-aware rate adaptation using PHY information for backscatter networks," in *Proc. IEEE Conf. Comput. Commun.*, Apr. 2018, pp. 1259–1267.

[40] W. Gong et al., "Channel-aware rate adaptation for backscatter networks," *IEEE/ACM Trans. Netw.*, vol. 26, no. 2, pp. 751–764, Apr. 2018.

[41] J. F. Ensworth and M. S. Reynolds, "BLE-backscatter: Ultralow-power IoT nodes compatible with Bluetooth 4.0 low energy (BLE) smartphones and tablets," *IEEE Trans. Microw. Theory Tech.*, vol. 65, no. 9, pp. 3360–3368, Sep. 2017.

[42] M. Zhang, J. Zhao, S. Chen, and W. Gong, "Reliable backscatter with commodity BLE," in *Proc. IEEE Conf. Comput. Commun.*, Jul. 2020, pp. 1291–1299.

[43] M. Zhang, S. Chen, J. Zhao, and W. Gong, "Commodity-level BLE backscatter," in *Proc. ACM MobiSys*, 2021, pp. 402–414.

[44] Y. Peng et al., "PLoRa: A passive long-range data network from ambient LoRa transmissions," in *Proc. ACM SIGCOMM*, 2018, pp. 147–160.

[45] V. Talla, M. Hessar, B. Kellogg, A. Najafi, J. Smith, and S. Gollakota, "LoRa backscatter: Enabling the vision of ubiquitous connectivity," in *Proc. ACM IMWUT*, 2017, pp. 1–24.

[46] A. Varshney, C. Pérez-Penichet, C. Rohner, and T. Voigt, "LoRea: A backscatter architecture that achieves a long communication range," in *Proc. ACM SenSys*, 2017, pp. 1–14.

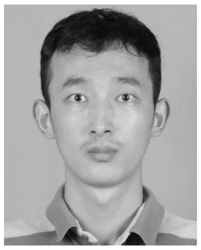
[47] J. Jiang, Z. Xu, F. Dang, and J. Wang, "Long-range ambient LoRa backscatter with parallel decoding," in *Proc. ACM MobiCom*, 2021, pp. 684–696.

[48] Q. Wang, S. Chen, J. Zhao, and W. Gong, "RapidRider: Efficient WiFi backscatter with uncontrolled ambient signals," in *Proc. IEEE Conf. Comput. Commun.*, May 2021, pp. 1–10.

[49] A. N. Parks, A. Liu, S. Gollakota, and J. R. Smith, "Turbocharging ambient backscatter communication," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 44, no. 4, pp. 619–630, 2014.

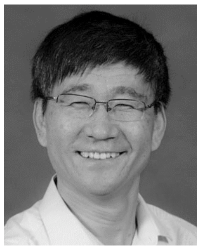
[50] V. Liu, A. Parks, V. Talla, S. Gollakota, D. Wetherall, and J. R. Smith, "Ambient backscatter: Wireless communication out of thin air," *ACM SIGCOMM*, vol. 43, no. 4, pp. 39–50, Aug. 2013.

[51] A. Wang, V. Iyer, V. Talla, J. R. Smith, and S. Gollakota, "FM backscatter: Enabling connected cities and smart fabrics," in *Proc. USENIX NSDI*, 2017, pp. 243–258.



Longzhi Yuan (Member, IEEE) received the B.S. and Ph.D. degrees from the University of Science and Technology of China, China, in 2017 and 2023, respectively. He is currently a Post-Doctoral Researcher with the Department of Computer Science, City University of Hong Kong. His research interests include wireless communication and sensing technologies.

Wei Gong (Senior Member, IEEE) received the B.S. degree from the Department of Computer Science and Technology, Huazhong University of Science and Technology, the M.S. degree from the School of Software, Tsinghua University, and the Ph.D. degree from the Department of Computer Science and Technology, Tsinghua University. He is currently a Professor with the School of Computer Science and Technology, University of Science and Technology of China. His research interests include backscatter networks, edge systems, and the IoT applications.



Yuguang Fang (Fellow, IEEE) received the M.S. degree from Qufu Normal University, the first Ph.D. degree from Case Western Reserve University, and the second Ph.D. degree from Boston University.

In 2000, he joined the Department of Electrical and Computer Engineering, University of Florida, as an Assistant Professor, then was promoted to an Associate Professor, a Full Professor, and a Distinguished Professor. Since 2022, he has been a Hong Kong Global STEM Scholar and the Chair Professor of Internet of Things with the Department of Computer Science, City University of Hong Kong. He is a Fellow of ACM and AAAS. He received many awards, including U.S. NSF CAREER Award, U.S. ONR Young Investigator Award, the 2018 IEEE Vehicular Technology Outstanding Service Award, the IEEE Communications Society AHSN Technical Achievement Award, the CISTC Technical Recognition Award, and the WTC Recognition Award. He was the Editor-in-Chief of IEEE TRANSACTIONS ON VEHICULAR TECHNOLOGY and IEEE WIRELESS COMMUNICATIONS.